

A Robust Visual Odometry and Precipice Detection System Using Consumer-grade Monocular Vision

Jason Campbell^{1,2}

Rahul Sukthankar^{1,2}

Illah Nourbakhsh^{2,3}

Aroon Pahwa²

¹Intel Research Pittsburgh
Pittsburgh, PA USA

²Carnegie Mellon University
Pittsburgh, PA USA

³NASA Ames Research Center
Moffett Field, CA USA

{jasoncam,rahuls,illah}@ri.cmu.edu aroon@cmu.edu

Abstract – We describe a monocular robot vision system which accomplishes accurate 3-DOF dead-reckoning, closed-loop motion control, and precipice and obstacle detection, all in dynamic environments, using a single, consumer-grade webcam and typical laptop computer hardware. Simultaneous translation and rotation are accurately measured, and the camera need not be placed at the robot's center of rotation. The algorithm is straightforward to implement and robust to noisy measurements. The software is based on open source computer vision libraries and is itself open source. It has been tested in a wide variety of real-world environments and on several different mobile robot platforms.

Index Terms - mobile robotics, mobile robot vision, optical flow, odometry

I. INTRODUCTION

Visual mobile robot navigation has long been a goal of robotics and computer vision researchers [9]. Whole fields have developed around the exploitation of one or more moving cameras, including visual simultaneous localization and mapping (V-SLAM) [8,21], visual servoing [12], spacecraft attitude control [14], and structure from motion (SFM). The past 15 years have also seen a smaller body of work on what has been termed visual odometry – that is, the incremental, online estimation of robot motion from a video sequence shot by an on-robot camera. [1,13,16,17,19,22].

Visual odometry is a distinctly local, low-latency approach that facilitates closed-loop motion control and obstacle and precipice detection, as well as highly accurate dead-reckoning. This local focus contrasts with more global approaches such as V-SLAM or online SFM which emphasize mapping, progressive reduction of uncertainty, global frames of reference, and which may include iterative refinement steps such as multi-frame bundle adjustment [10].

Historically, visual odometry systems have had difficulty overcoming a number of problems, including numerical instabilities common in SFM-like projective geometric techniques [7], sensitivity to the low quality of point correspondences available from automatic tracking algorithms [2,23], a requirement for omnidirectional views [13,22], or the inability to disambiguate simultaneous rotation and translation [11]. A more recent result overcomes these difficulties in monocular- and stereo-camera cases but achieves high accuracy only with a calibrated stereo pair [19].

In this paper we describe an algorithm for strictly monocular visual odometry which achieves good real-world performance (similar to the calibrated-stereo results shown in [19]) at substantially lower implementation complexity. This is made possible by exploiting a variety of straightforward consensus methods, several reasonable assumptions

for ground-based mobile robots, and the increasingly high performance of consumer computing hardware. We have tested this system on three small mobile robot platforms (see Figs. 1 and 5), and in traditional office environments as well as in extreme terrain such as ice. Our visual odometry system is readily accessible to a wide range of robot builders because it is based on and is itself open source software.

Visual odometry offers the prospect of substantially reduced sensing costs, allowing more reliable navigation through unstructured areas and safer operation in close proximity to humans. As the cost of computation falls, an inexpensive camera can replace a typical sensor suite consisting of dozens of range sensors and a set of encoders and provide a broader field of view and the ability to perform range and appearance-based sensing simultaneously. Passive vision systems also avoid the multi-path interference problems typical with sonar rangefinders and the high sensitivity to lighting common in low-cost infrared rangefinders. For kinematically indeterminate robots (e.g., where friction is low and actuation powerful), visual odometry offers a low-latency error signal which can be used in a feedback loop to correct motion. For robots that operate in highly unstructured indoor environments (e.g., urban search and rescue) visual odometry can be significantly more practical and functional than other localization systems because no radio coverage is required, no beacons need be carried/deployed, static drift is low compared to low-cost inertial measurement units, and high degrees of wheel slip pose no difficulty.

The remainder of this paper is organized as follows: Section II describes the hardware used in our experiments. Section III outlines the visual odometry algorithm itself. Sections IV through VI describe detailed aspects of the al-

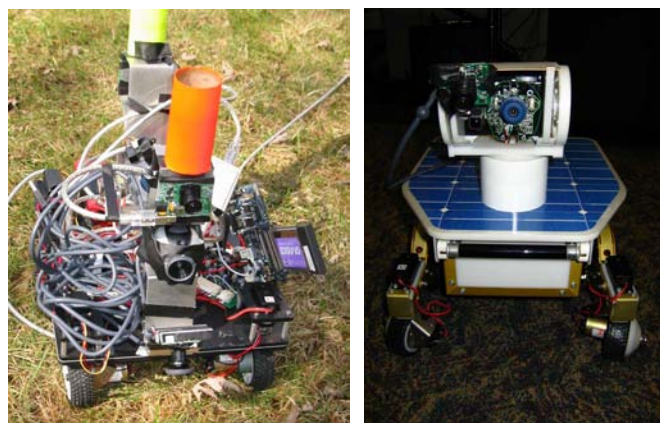


Fig. 1 Two examples of robots using this visual odometry system
Only the uppermost, front camera on each robot is used for visual odometry.

gorithm. Section VII addresses obstacle and precipice detection. Section VIII summarizes the framework within which we evaluate this and other visual odometry systems. Section IX describes the closed-loop version of our vision system. Section X discusses the real-world performance of our algorithm in a variety of environments and on a variety of robots, and Section XI explores the implications of robust visual odometry for the design of new robot systems. Section XII concludes and summarizes our findings.

II. HARDWARE CONFIGURATION

We have successfully deployed our visual odometry system using a variety of cameras, ranging from \$5 surplus webcams to multi-megapixel consumer digicams. The results we report here are from USB- and IEEE1394- webcams operating at VGA resolution (640x480 pixels). These cameras, fitted with inexpensive 2.1 mm wide-angle lenses, have unit costs around \$100 and achieve peak frame rates of 10 and 30 frames per second (fps), respectively. Each camera affords a horizontal field of view of approximately 90° . In the USB case, a proprietary compression algorithm was required between the camera and camera driver [18] to permit 10 fps VGA operation over a USB 1.1 bus.

Two types of tests were performed, one that passively interpreted video to compute the robot's trajectory after-the-fact ("open loop"), and another that used on-line visual odometry estimates to actively correct the robot's trajectory ("closed loop"). The open loop tests were performed at 7.5 fps using uncompressed video recorded by an IEEE1394-attached webcam. The open loop vision system computations were run off-line on a 3.2 GHz Intel® Pentium® Xeon™ desktop PC at effective frame rates between 30 and 50 fps. The closed loop tests were performed using a USB-attached webcam capable of up to 10 fps, however, the actual frame rate varied due to CPU load. Using a laptop equipped with a 1.5 GHz Intel® Pentium® M processor for vision processing, robot control, and a near-real-time graphical user interface displaying the estimated optical flow field and resulting visual odometry results, rates from 2 to 10 fps were observed.

III. ALGORITHM OVERVIEW

The visual odometry algorithm we present here is an 8-stage process where all eight stages execute sequentially after the capture of each video frame. In practice this algorithm can execute as quickly as 15 fps on common laptop hardware. Our approach relies on two assumptions which are quite reasonable for many mobile robots: a) the robot travels predominantly over a ground plane, and b) the camera is mounted rigidly with respect to the robot and ground plane. The quality of position estimation results varies with frame rate, robot speed, camera height, and visual environment. This section presents a brief outline of the eight stages and the following sections address specific stages in detail. In this discussion we shall use the notation (u,v) to represent image coordinates, lowercase (x,y,θ) to represent incremental coordinates in a robot-centered frame of reference, and uppercase (X,Y,Θ) to represent global coordinates in a frame of reference based on the starting position and orientation of the robot. The eight stages are:

1. Correct camera images for lens distortion. This is accomplished using standard computer vision algorithms included in the Open Computer Vision Library (OpenCV) [5].
2. Estimate the optical flow field corresponding to recent video frames.
3. Screen flow-field vectors for potential tracking errors and independently moving objects. Discard such vectors.
4. Divide the optical flow field into "ground" and "sky".
5. Project image coordinates (u,v) of flow vectors in the "sky" region into a robot-centered cylindrical coordinate frame and determine consensus angular displacement between the last two camera views. Use this as the estimate θ of rotational motion.
6. Project image coordinates (u,v) of flow vectors from the "ground" region on to ground plane coordinates. From this set of transformed vectors, determine a consensus x - y feature displacement between the last two camera views and use this as the estimate (x,y) of translational motion.
7. Sum incremental measurements (x,y,θ) to arrive at a cumulative estimate (X,Y,Θ) of robot position relative to the robot's original starting position.
8. Periodically repopulate the set of active trackpoints to maintain adequate and uniform coverage across ground and sky.

IV. ESTIMATING THE OPTICAL FLOW FIELD

Optical flow field estimation remains an open problem after over 25 years of effort. Dozens of methods have been proposed, each with its own strengths and weaknesses. In a 1995 survey [2,3], Barron, Fleet, *et al.* reported the Lucas Kanade method [15] as being among the most accurate and most reliable of the methods (then available), when tested using a variety of simulated and real data series. An improved and more efficient form [4] of the Lucas Kanade algorithm is available in the open source computer vision library, OpenCV. We employ this feature tracker to obtain the point correspondences (equivalently, the optical flow field vectors) required in subsequent stages of the algorithm.

In selecting features to track, we use a low corner threshold and specify a large minimum distance between features. (typically 0.005 and 20 pixels, respectively) This encourages the detection of features even in low-contrast parts of the image and provides relatively uniform coverage across the field of view, at the cost of generating a higher number of tracking errors. Given that the robust methods we use to analyze those feature tracks can deal with substantial numbers of outliers, we have found such wider coverage preferable to minimizing tracking errors.

V. SCREENING TRACKPOINTS FOR QUALITY

Low contrast images, feature-poor environments, occlusions, and spatial aliasing may all cause individual optical flow vectors to be estimated incorrectly (henceforth, "patch tracking errors"). Also, dynamic elements in the environment (e.g., moving people, vehicles) can induce optical flow that may be substantially at odds with the robot's ego-motion-induced optical flow. Our visual odometry system attempts to mitigate the impact of confusing flow vectors from patch tracking errors and independently moving

objects by identifying and excluding suspect vectors from the ego-motion estimation process.

Patch-tracking errors tend to show erratic movement from frame to frame. For instance, at occlusion boundaries, features being tracked disappear or change due to changing image content near the discontinuity. Low-contrast features may simply be difficult to precisely relocalize in a subsequent video frame due to spatial sampling irregularities inherent to typical low cost image sensors. Specular reflections may be tracked as features and shortly thereafter disappear due to surface conditions and angles.

Likewise, the optical flow field associated with dynamic elements in the environment also tends to show more erratic motion over time because the flow vectors created by moving objects are superimposed on to the flow generated by the robot's ego-motion. In a special, but very important case, the optical flow field associated with moving humans tends to be particularly erratic because human motion is typically discontinuous and moving human bodies undergo substantial deformation.

In contrast, the optical flow field due to ego-motion tends to be smooth because the robot's motion is largely smooth. (While there exist a variety of robots that exhibit discontinuous motion, such as legged mechanisms, a smooth motion assumption is reasonable for most mobile robots. Furthermore, a robot which exhibits periodic or known discontinuous motion could actually exploit such motion to better disambiguate the optical flow field.) Thus, for both tracking errors and independent motion, smoothness can be employed as a practical heuristic to assess the reliability of each optical flow track in estimating robot motion. Our optical flow algorithm exploits this observation by focusing attention on elements that exhibit smooth motion over a seven-frame (0.5-3 second) time horizon. However, to account for cases where the robot's actual motion is temporarily not smooth (e.g., during travel over a bumpy surface), we suspend this rule in cases where the majority of vectors exhibit unsmooth motion over the observation interval.

The criteria used to identify "smooth" flow centers around the incremental angles between successive flow vectors associated with each tracked image patch over time. We compute the flow-field direction at the patch in question across three intervals – seven to three frames back, three to one frames back and one frame back to the current frame – and then classify an element as "unsmooth" when the difference between any of those directions exceeds a threshold of 30 degrees. Once an element has been classified as unsmooth, it is ignored for purposes of egomotion estimation within the current cycle.

Over time we also wish to stop tracking consistently unsmooth elements and instead choose other image patches which may offer a more reliable indication of robot egomotion. Thus, when a given flow element is classified as "unsmooth" but the majority of other flow elements are classified as smooth, a score field associated with the unsmooth element is incremented by 5. The score fields for all smooth elements are decremented by 1, but not below zero. If, over time, a flow element's unsmoothness-score field exceeds 10, that element is removed from the list of image patches to track in future video frames. This allows problematic patches to be replaced with new image patches

the next time that new tracking features are chosen (i.e., in Step 8). We have found that the weights 10/5/1 strike a compromise between shielding the motion estimation process from confusing optical flow vectors and accommodating a limited number of irregularities without prematurely discarding a trackpoint.

VI. ESTIMATING ROBOT MOTION

Given the set of filtered optical flow vectors, the next task is to derive an estimate of the robot's incremental motion during the last pair of frames, and to integrate it over time to obtain global estimate of robot position. First, we summarize the derivation of the mapping between image pixels and points in the world. Next, we decompose incremental motion into a rotation and a translation and estimate each separately. Finally, we combine the information to recover the robot's motion in a world coordinate frame. These steps are detailed in the following subsections.

A. Mapping Camera Coordinates to the Ground Plane

First, we determine the mapping between the camera coordinates (u,v) of a tracked feature and its corresponding point (x,y) on the ground plane. Our physical setup is depicted in Fig. 2. The height H of the camera from the ground plane is manually measured, as is the distance D from the robot to the intersection of the principal ray with the ground plane. We can recover the tilt α of the camera from the expression:

$$\tan(\alpha) = H/D \quad (1)$$

We also employ basic trigonometry to map from v (assumed to be measured from the top of the image) to β (vertical angle between principal point and the observed point):

$$\tan(\beta) = (2v - V) \tan\left(\frac{VFOV}{2}\right), \quad (2)$$

where V is the vertical dimension of the image in pixels (e.g., 480 rows) and $VFOV$ is the vertical field of view of the camera. These relations enable us to recover the distance y from the robot to the observed point:

$$y = \frac{H}{\tan(\alpha + \beta)}, \quad (3)$$

and the depth z to the observed point from the camera:

$$z = \frac{H \cos(\beta)}{\sin(\alpha + \beta)}. \quad (4)$$

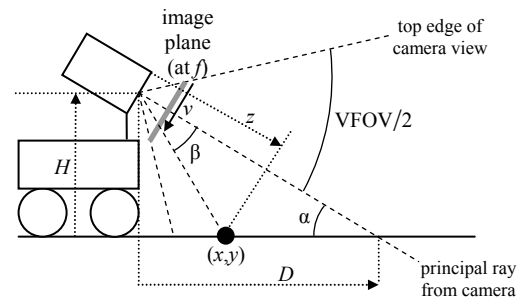


Fig. 2 Mapping Camera Coordinates to the Ground Plane

When an optical flow vector generated by a tracked feature on the ground plane is imaged by the camera, its observed magnitude (in pixels) is affected by its depth z relative to the image plane and its orientation β to the camera axis. The above equations enable us to invert this transform, and to determine the actual displacement of each observed feature on the ground plane – and thus derive the incremental motion of the robot.

B. Estimating Robot Rotation and Translation

Over a short interval, the robot’s motion on the ground plane can be decomposed into a change in heading (rotation about a vertical axis) and a displacement (translation). We recover each separately using different regions of the image. Such a two-step process is important because, while in principle all optical flow vectors can convey information about camera translation, vectors corresponding to features that are distant from the camera will exhibit such small amounts of parallax-induced optical flow that those measurements would be overwhelmed by tracking noise. In contrast, camera rotation causes all observed feature points, both nearby and distant, to move through the same angle. This means that we can derive a robust estimate for heading change by observing features far from the camera (since those are relatively insensitive to optical flow induced by translation). Once the effects of any observed heading change are subtracted from the optical flow field, the flow vectors associated with features near the camera can serve as a good basis for estimating translational motion. We implement this strategy by observing that flow vectors near (and above) the horizon typically correspond to distant objects and we employ these for rotation estimation. Vectors below the horizon are good candidates for estimating translation. Fig. 3 shows a typical image with filtered flow vectors and its decomposition into the three zones. We discard vectors in a “dead zone” near the horizon since the observed location of the horizon is affected by brief, transient changes in robot pitch – and this could greatly magnify small errors in translation estimates based on distant points (which lie near the horizon). Once again, because further steps in the algorithm are robust against outliers, it is not necessary that this sky/horizon/ground heuristic always hold true. For instance, we have observed good visual odometry performance even in crowded rooms, where many features above the horizon are in fact only tens of centimeters or a few meters from the camera.

To estimate rotation, we back-project each flow vector from the “distant” zone into a vertical cylindrical coordinate frame centered on the camera using mappings analogous to those described above. We then simply estimate the change in heading by taking the median of the observed angular displacements. Note that one should not naively use the horizontal displacement of the flow vector as an estimate for heading change since, particularly for wide-angle lenses, the mapping from the camera plane to cylindrical coordinates will depend on the position of the observed feature in the image. However, for certain low-cost, highly-barrel-distorting wide-angle lenses (such as the ones used in most of our tests), we have empirically observed that the barrel distortion fortuitously approximates the cylindrical mapping. Thus one could achieve comparable accuracy with a

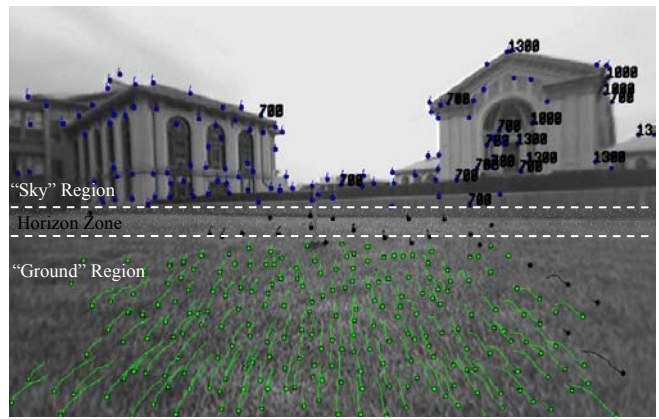


Fig. 3 Classifying Optical Flow Field Vectors

Vectors above the horizon are used to estimate robot rotation, vectors below the horizon are used to estimate robot translation.

significant savings in computation simply by performing optical flow in the uncorrected image and using the median of horizontal displacements as an indicator of heading change. While our present algorithm does not take advantage of this trick, we are developing an extension that will be able to do so. Obviously, this trick cannot be employed when estimating translation because the transformation required is entirely different.

To estimate translation, we first subtract the rotational flow field implied by our estimated heading change. We then attempt to find the pure translation vector that best explains the resulting flow observed in the “ground” region of the image. This is done by back-projecting the flow vectors on to the ground plane as outlined in (3). This inverts the perspective distortion induced by the camera, and the lengths of each vector correspond to actual displacements on the ground plane. For an ideal image sequence, the resulting flow field would consist of identical vectors; in reality, the flow field is perturbed by unmodeled effects such as tracking error, variations in the height of the ground surface, and the motion of other objects in the scene. We obtain a robust estimate of the robot’s translation by taking the median of the set of x -displacements and the median of the set of y -displacements. Such a simple scheme works because we have already accounted for the effects of perspective projection and rotation. Note that this simplicity is possible due to the strong (but valid) assumptions described above.

The median x displacement serves as our estimate of instantaneous translation along the focal plane (i.e., sideways motion) and the median y displacement serves as our estimate of instantaneous translation parallel along the camera axis (i.e., forward/reverse motion). Or in the case where the camera is tilted these two axes are defined by the projections onto the ground plane of the focal plane and the camera axis.

C. Estimating Global Motion

The previous subsection described our algorithm’s approach for estimating incremental robot motion using a consensus of appropriately back-projected optical flow vectors. These incremental changes in robot pose are chained frame-by-frame to derive the global estimate of the robot’s position.

VII. DETECTING HAZARDS

In addition to accurately estimating translation and rotation of the robot in a planar environment, our system exploits optical flow to identify potential hazards, such as obstacles and precipices. In contrast to traditional obstacle-detection techniques, we do not recover a depth map of the scene, nor do we rely on the reflectance properties of obstacles. The basic idea is straightforward: discontinuities in the optical flow field signal the presence of both obstacles (positive violations of the planar world hypothesis) and precipices (negative violations). Our current approach to detecting these discontinuities relies on somewhat *ad hoc* heuristics. The image plane is divided into several sub-regions, each of which independently computes the median optical flow field direction and velocity. Adjacent sub-regions are then compared for evidence of a discontinuity.

Since obstacles or precipices are, by definition, located at a different depth from the camera, the observed flow due to parallax will be different. A region with disproportionately-short flow vectors signals a depression while a region with longer vectors is indicative of a looming obstacle. Thus, when our algorithm notices a sudden change in the median value of flow vectors in a region, it signals the presence of a hazard. The robot can monitor the location of hazards as it continues its careful advance, and can safely stop before the hazard becomes a threat. This is illustrated in Fig. 4.

In practice, our robots can safely detect precipices as near as 3cm away and can autonomously maneuver on small table-tops (approximately 1 m²) without falling.

VIII. MEASURING THE PERFORMANCE OF A VISUAL ODOMETRY SYSTEM

A visual odometry system may be used in two modes: open loop (i.e., passive observation of the robot's motion), or closed loop (i.e., integrated with the robot's motion control logic as a feedback sensor). Closed-loop operation is more useful and ultimately more likely in practice, but open loop testing is substantially more informative when evaluating a visual odometry system. This is because in certain modes of closed loop operation (e.g., whenever velocity in one dimension or one axis is being held at zero), a wide variety of control loop gain values may yield satisfactory performance. This flexibility corresponds to insensitivity to an unknown scale factor in estimated change in position or orientation. By contrast, open loop evaluation highlights bias errors in translational and angular velocity estimation that manifest as cumulative error.

In the following evaluation we use a combination of open loop and closed loop methods, including open loop experiments patterned after the procedure we outlined in [6] and closed loop evaluations versus other forms of dead reckoning. To measure the relative benefit of open loop visual odometry we consider three types of position estimation error (also as proposed in [6]), including a) incremental translational error, b) incremental rotational error, and c) long-run net (cumulative) translational error. In the case of (c) we express this error as a percentage of total distance traveled to enable comparisons between test scenarios done

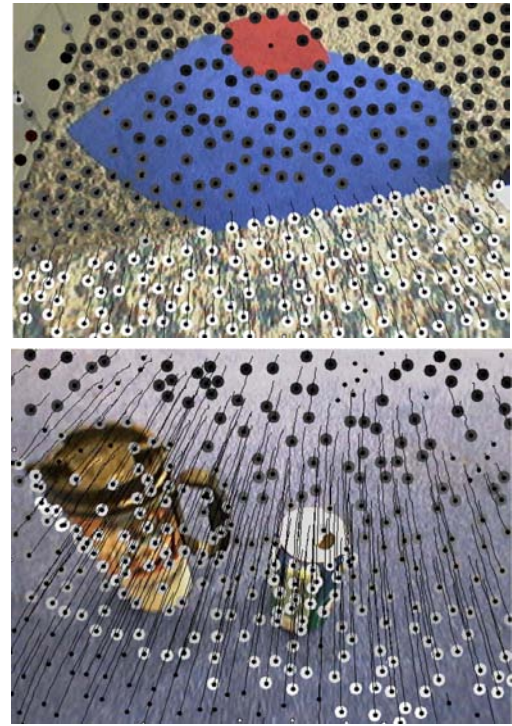


Fig. 4 Hazard detection

The upper image shows a precipice, lower image shows a pair of obstacles. Each shaded circle indicates the relative optical flow field velocity at that point. Lighter circles denote higher velocities and darker circles denote lower velocities.

at different scales. To measure these we use several ground-truthed visual odometry video sequences.

In the closed loop analysis we measure the net Euclidean distance between the final robot position and the goal position at the end of a commanded path. We then divide this distance by the total distance traveled to obtain a basic measure of cumulative error rate in translational motion. Each of the robot paths involved around 1000 frames of video taken over 4-10 m of forward travel and included at least 360° of cumulative rotation to test the ability to correctly chain estimates derived from different sets of features.

For both open loop and closed loop tests we use real imagery obtained from on-robot cameras. While real imagery is somewhat more difficult to acquire and ground-truth, simulated data does a poor job of capturing a host of factors which can have a significant impact on vision system and robot performance, including variable lighting, motion blur, optical imperfections, complex motions, and temporal and spatial aliasing. The complex interplay of these factors also offers strong incentive to test robot vision systems in as wide a variety of environments and lighting conditions as possible.

IX. CLOSING THE LOOP

To implement the closed loop tests we developed a simple proportional controller for one of our robots. This controller uses visual odometry to estimate robot position relative to the starting point and attempts to execute compound motion plans consisting of sequences of pure translation and pure rotation. Although the motion plan is constructed from segments of pure translation and rotation, the controller employs mixed translation and rotation to

smoothly correct errors observed during execution. This allows the controller to appropriately counteract steady-state drift due to differential drag forces, uneven tire wear, and drivetrain irregularities, as well as transient conditions such as loss of traction and deliberate human interference. Specifically, the controller maintains 3DOF estimates of the robot's pose error and adjusts motor drive speeds and steering angles to correct those errors.

Given the number of stages involved in obtaining the video images, processing them, and commanding the robot, the total latency of the control loop is relatively large – approximately 300 ms. This leads to substantial overshoot on each motion segment involved in executing a motion plan. Rather than improving the controller to get better single-movement accuracy we instead use the vision system to measure these additional errors and factor them into the execution of the next motion segment. Thus we take advantage of strong visual dead-reckoning performance to counteract the failings of both a naïve controller and kinematic indeterminacy (e.g., wheel slip, experimenter interference, surface irregularities).

X. RESULTS

We have integrated the closed loop version of the visual odometry system described above into a mobile robot system based on the Personal Exploration Rover (PER) platform [20]. The vision system itself runs on a laptop computer which sits adjacent to the robot and is connected via a tether. This mode of operation is most appropriate for demonstrations and interactive experiments where accessibility to the user interface displayed on the laptop is important. Via this interface the user can control vision system parameters such as number of trackpoints, image patch size, and required corner quality. The laptop can also be attached to the back of the robot for longer-distance experiments where tethered operation is impractical or when access to the user interface is unnecessary. This mobile robot vision system has proven effective in a variety of difficult conditions such as polished floors (challenging due to specular reflections), and crowded rooms full of humans (challenging due to the many independently moving entities in view).

We have also evaluated our algorithm using prerecorded video data where ground-truth has been established with the fiducial-based robot tracking system we described in [6]. This has allowed us to evaluate the performance of our algorithm in a variety of outdoor and indoor situations, including such comparatively extreme cases as ice, high-glare asphalt and grass. Results are presented in Table I. For comparison, consider that a recently published result in visual odometry using *calibrated stereo cameras* on a ground vehicle yielded corresponding cumulative (open loop) error measures of 3.6% and 4.6% on two long test runs [19]. Our proposed visual odometry algorithm makes more restrictive assumptions but uses only a monocular camera and achieves similar error rates while being substantially less complex to implement. While our tests involve substantially shorter distances than those in [19], the relative error measurements and motion topologies are comparable.

TABLE I
VISUAL ODOMETRY PERFORMANCE ON VARYING TERRAIN

Terrain (any special circumstances)	Mean Incremental Error Magni- tude	Cumulative Error, (Cartesian Distance as % Of Distance Traveled)
Open-loop tests		
Indoors / Carpet	0.3	3.3%
Outdoors / Grass	2.2	5.1%
Outdoors / Asphalt	4.3	6.1%
Outdoors / Ice	3.5	5.4%
Closed-loop tests		
Indoors / Polished Concrete (with active, physical human interference and an independently moving audience)	unmeasured	7.1%

XI. IMPLICATIONS FOR ROBOT DESIGN

Experimental results validate our contention that consumer-grade cameras offer sufficient perceptual information to enable effective visual odometry and precipice detection. Such cameras offer a new price / functionality point in the space of possible robot sensors.

We believe the present results further the case for vision-based mobile robotics, and also suggest directions for robot morphology. Visual odometry requires a large camera height concordant with both visibility of the terrain over the robot itself and to facilitate measurement of optical flow for both nearby and distant features. Placement of the camera at the robot's center of rotation is often in tension with this desire to push the camera forward so that the terrain in front of the robot is visible. Our visual odometry results suggest that, even with the camera at a significant distance from the robot center of rotation, odometry remains accurate, and so the camera can be placed to optimize forward and downward visibility. Furthermore, we believe that holonomic robots and other systems capable of controlling robot motion along any arbitrary center of rotation will be able to use this VO technique to measure position and control trajectory. This is particularly important in the case of holonomic robots using Swedish 90 wheels, such as the Palm Pilot Robot Kit (PPRK), because such wheels tend to have significantly higher wheel-ground slippage than conventional fixed or steerable wheels. In effect, the visual odometry algorithm described here, combined with consumer-grade vision hardware, imbues the robot designer with the freedom to design and deploy robots with more expressive motion regimes because this technique obviates the need for accurate wheel encoders and wheel-ground slip constraints.

Cameras are easier to place than sonars or IRs because they image a broad field of view with relatively high resolution. With adequate robustness in the VO system, cameras need not be center-mounted, increasing their flexibility and offering more opportunities to place them ideally for precipice detection. Given that modern robots are very likely to incorporate high-performance processors typical of today's PCs and laptops a vision system can readily be implemented without the need for additional hardware. PTZ cameras can serve several navigational configurations for omnidirectional robots well.

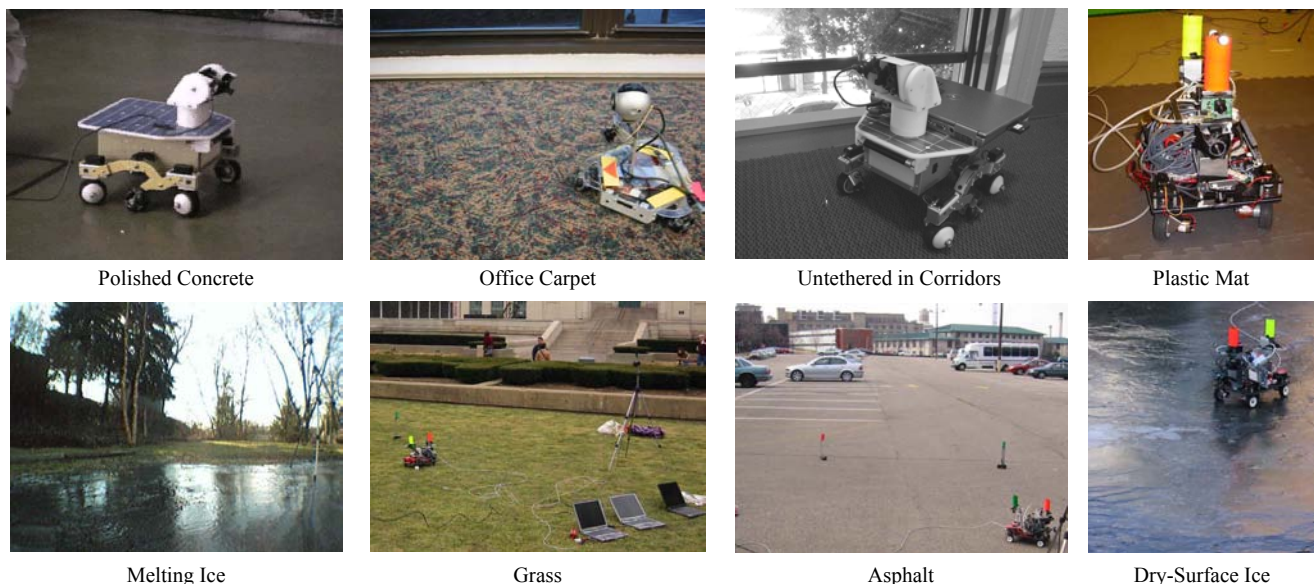


Fig. 5 Environments and Robots On/In Which This Visual Odometry System Has Been Tested

XII. CONCLUSION

Vision is falling in price more rapidly than any other sensor, and yet is also a richer sensor than traditional ranging devices. In particular because a camera can capture so much data simultaneously, even a monocular vision system can play multiple roles within a robot system. This paper has described a vision system capable of robustly estimating robot velocity and rotation as well as reliably detecting certain classes of hazards (precipices and obstacles). This vision system is readily accessible to all levels of robot builders, requiring only a single consumer-quality webcam. We have evaluated the proposed system on a variety of small mobile robots and tested it in a wide variety of environments (see Fig. 5). Our implementation of this algorithm is available as open source software and can be downloaded from

<http://info.pittsburgh.intel-research.net/People/jasonc/vo>.

ACKNOWLEDGMENT

We wish to thank Emily Hamner, Eric Porter, Brian Dunlavey and the CMU Personal Rover Project for their supply of a Personal Exploration Rover for this work. Aroon Pahwa was supported by the Intel First Year Research Experience (IFYRE) program at CMU.

REFERENCES

- [1] O. Amidi, T. Kanade, and J. Miller. "Vision-Based Autonomous Helicopter Research at CMU." *Proceedings of Heli Japan*, 1998.
- [2] J. Barron, D. Fleet, S. Beauchemin, and T. Burkitt, "Performance of Optical Flow Techniques," *International Journal of Computer Vision*, vol. 10, pp. 43-77, 1994.
- [3] S. Beauchemin and J. Barron. "The Computation of Optical Flow," *ACM Computing Surveys*, 27(3), pp. 433-467, 1995.
- [4] J.-Y. Bouguet. "Pyramidal Implementation of the Lucas Kanade Feature Tracker," OpenCV Documentation, Intel Corporation, Microprocessor Research Labs, 1999.
- [5] G. Bradski, et. al. Intel Open Source Computer Vision Library: <http://www.intel.com/research/mrl/research/opencv/>
- [6] J. Campbell, R. Sukthankar, I. Nourbakhsh. "Techniques for Evaluating Optical Flow in Extreme Terrain," *Proceedings of IROS 2004*.
- [7] R. Cipolla, Y. Okamoto, and Y. Kuno. "Robust Structure From Motion Using Motion Parallax," in *Proceedings of IEEE ICCV*, 1993.
- [8] A. Davison, Y. Cid, N. Kita. "Real-Time 3D SLAM with Wide-Angle Vision," in *Proceedings of IFAC Symposium on Intelligent Autonomous Vehicles*, 2004.
- [9] D. Gennery and H. Moravec, *Cart Project Progress Report*, Stanford University, contract NASW 2916, Sept 1976.
- [10] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
- [11] B. Horn and E. Weldon. "Direct Methods for Recovering Motion," *International Journal of Computer Vision*, vol. 2, pp. 51-76, 1988.
- [12] S. Hutchinson, G. Hager, and P. Corke. "A Tutorial on Visual Servo Control," *IEEE Trans. Robotics and Automation*, vol. 12, no. 5, pp. 651-670, Oct 1996.
- [13] A. Levin, R. Szeliski. "Visual Odometry and Map Correlation," in *Proceedings of CVPR 2004*.
- [14] C. Liebe. "Star Trackers for Attitude Determination," *IEEE Aerospace and Electronic Systems Magazine*, vol. 10, no. 6, pp. 10-16, June 1995.
- [15] B. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision," *International Joint Conference on Artificial Intelligence*, pp. 674-679, 1981.
- [16] R. Marks, H. Wang, M. Lee, and S. Rock. "Automatic Visual Station Keeping of an Underwater Robot," in Volume 2, *Proceedings of IEEE Oceans '94*, pp. 137-142.
- [17] S. Negahdaripour and B. Horn. "Direct Passive Navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 168-176, January 1987.
- [18] Nemosoft Webcam driver: <http://www.smcc.demon.nl/webcam>
- [19] D. Nistér, O. Naroditsky, J. Bergen. "Visual Odometry," in *Proceedings of CVPR 2004*.
- [20] Personal Rover Project: <http://www.cs.cmu.edu/~personalrover>
- [21] S. Se, D. Lowe, J. Little. "Vision-Based Mobile Robot Localization and Mapping Using Scale-Invariant Features," in *Proceedings of ICRA 2001*.
- [22] O. Shakernia, R. Vidal, S. Sastry. "Infinitesimal Motion Estimation from Multiple Central Panoramic Views," in *Proceedings of IEEE Workshop on Motion and Video Computing*, December 2002.
- [23] C. Tomasi and J. Zhang. "Is Structure-From-Motion Worth Pursuing?" *Seventh International Symposium on Robotics Research*, ISRR '95, October 1995.